Title: Routing and Rerouting in Territorial Systems Modeled by Weakly Dynamic Graphs

Authors: Moustafa NAKECHBANDI[(1)], Jean-Yves COLIN[(1)], Ahmed Salem OULD CHEIKH[(1, 2)],

(1) LITIS, Le Havre University, 5 rue Ph. Lebon, BP 540, 76058, Le Havre, France,

{moustafa.nakechbandi, jean-yves.colin}@univ-lehavre.fr.

(2) Nouakchott University, Nouakchott, Mauritania. ahdsalem@univ-nkc.mr

Keywords: Dynamic Graph; Shortest Path Problem; Road Network; Route Planning; Intelligent Transport Systems.

Extended abstract.

Static graphs have a long history of being used to efficiently represent static problems. In these problems, all the data are known from the start. The real world is not static, however, and the solutions to static problems may not always be used [13, 2]. Some data may change, or be unknown in advance. In territorial systems, for example, the traversal duration of a location may depend on traffic density, the presence or not of traffic jams, work in progress, etc. that are all time dependent and usually hard to predict. Thus several approaches have been proposed to study parametric graphs [1] and dynamic graphs [6].

Fully dynamic algorithms, for example, are applied to problems that can be solved in polynomial time. They start with a computed optimal solution, and then try to maintain them when changes occur in the problem. They often propose sophisticated data structures to reach this goal [8, 11].

When the delay between a change and the moment a new solution is needed is very small, or when the problem itself is NP-hard, faster algorithms are needed These reoptimizing algorithms usually start from an initial solution that is not optimal but is expected to be of good quality, if possible. As soon a change is detected, they compute a new solution, trying to do it faster that classical algorithms. Or they compute a new solution as fast as the classical algorithms but this resulting solution is better than the ones found by classical algorithms. These algorithms include meta-heuristics such as ants colony algorithms [5], or swarm algorithms [3].

Another approach used is probabilistic. Probabilities are associated to some variables in the graph, such as the value of a weight, or the presence of a vertex or of a constraint, for example. The algorithms used in these problems usually compute a solution then do some robustness analysis in the probability space [9]. Or they do a quick re-optimization of the solution once the parameters of the problem are perfectly known [4, 10].

In this paper, we study weakly dynamic directed acyclic graphs. In these weakly dynamic graphs with positive weighted arcs, one or two arcs are known to be non stable. That is, the weight of each of these non stable arcs may change at any time. All other arcs have stable weights that never change. We are interested in the “One-to-All” shortest path problem (SPP), that is, in finding what are the shortest paths from one vertex to all other vertices of this graph. This must be done considering the weights of the non stable arcs. Preliminary results on weakly dynamic graphs with only one variable arc or edge were presented in [12, 7]. Here, we propose an efficient algorithm that pre-computes alternative shortest paths for all possible values of the variable weights. It also computes very small sets of very simple critical conditions. When the non stable weights change, the shortest paths for these new values may then directly and immediately be deduced from the critical conditions and may immediately be used without any further recomputation.

We consider a directed acyclic graph (DAG) G=(V,E). *V* is the set of vertices, *E* is the set of arcs, and to each arc is associated a positive weight. Almost all the arcs are stable and their weights never change. However two known arcs are not stable and their weights may change at any time. The first variable arc is denoted $(x^1{}_1, x^1{}_2)$, and its variable weight is x^1. The second variable arc is denoted $(x^2{}_1, x^2{}_2)$, and its variable weight is x^2. We call this kind of DAG a weakly dynamic graph with two known variable arcs.

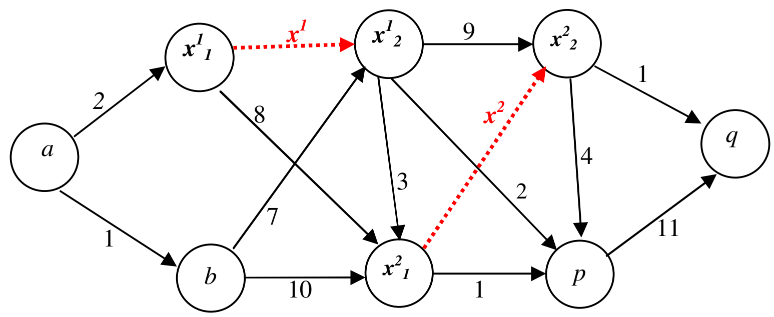

Fig. 1 – *Example of weakly dynamic graph with two variable arcs (in red on the graph).*

The length of a path is the sum of the weights of its arcs. Shortest paths that do not include any variable arc may be computed with Dijkstra algorithm.

The proposed algorithm works in four steps

1. it computes the shortest paths that do not include any variable arc, from the starting vertex to all other vertices,
2. for each starting vertex $x^r{}_1$ of a variable arc $(x^r{}_1, x^r{}_2)$ that was reached during step 1, it computes again shortest paths that do not include any variable arc, from the other vertex $x^r{}_2$ of the variable arc to all other vertices,
3. for each starting vertex $x^s{}_1$ of a variable arc $(x^s{}_1, x^s{}_2)$ that was reached during step 2, it computes again shortest paths that do not include any variable arc, from the other vertex $x^s{}_2$ of the variable arc to all other vertices,
4. for each vertex, it finally compares up to 4 values
    a. the constant length of the shortest path that does not include any variable arc,
    b. the variable length of the shortest path that includes only the first variable arc,
    c. the variable length of the shortest path that includes only the second variable arc,
    d. the variable length of the shortest path that includes the two variable arc.

When comparing the four lengths, some of them have the variable values of the variable arcs as parameters. Thus, for a given vertex, it may happen that one path from the starting vertex may be better than another path for some particular values of the variable weights, and may be worse for some other values of the variable weights.

**Example : ** We now apply this algorithm on the graph of Fig. 1.

Step1: Here we use Dijkstra's algorithm to computed shortest paths that do not include any variable edge from a (origin) to all other vertices, the short distance will be noted $ds^0(a, j)$:

Vertex j	$\boldsymbol{a}$ (origine)	b	x^1_1	x^1_2	x^2_1	x^2_2	p	q
$ds^0(a, j)$	0	1	2	8	10	17	10	18

Step2: Computes the shortest paths that do not include any variable arc, from the starting vertex x^r_1, x^r_2 of a variable arc to all other vertices :

Vertex j	$\boldsymbol{a}$	b	x^1_1	x^1_2	$\boldsymbol{x^2_1}$	$\boldsymbol{x^2_2}$	p	q
$ds^1(x^1_2, j)$: *shortest paths from the starting vertex of variable edge* x^1_2 *to all other*	-	-	-	0	3	9	2	10
$ds^1(x^2_2, j)$: *shortest paths from the starting vertex of variable edge* x^2_2 *to all other*	-	-	-	-	-	-	4	1
$ds^0(i, x^1_1) + x^1 + ds^1(x^1_2, j)$	-	-	-	$2+x^1$	$2+x^1+3$	$2+x^1+9$	$2+x^1+2$	$2+x^1+10$
$ds^0(i, x^2_1) + x^2 + ds^1(x^2_2, j)$	-	-	-	-	-	$10+x^2$	$10+x^2+4$	$10+x^2+1$

Step3: Compute $ds^0(i, x^1_1) + x^1 + ds^1(x^1_2, x^2_1) + x^2 + ds^2(x^2_2, j)$ *for all vertex j*

Vertex j	$\boldsymbol{a}$	b	x^1_1	x^1_2	x^2_1	x^2_2	p	q
$ds^0(i, x^1_1) + x^1 + ds^1(x^1_2, x^2_1) + x^2 + ds^2(x^2_2, j)$	-	-	-	-	-	$2+x^1+3+x^2$	$2+x^1+3+x^2+4$	$2+x^1+3+x^2+1$

Step4: Compare the 4 values computed in the last steps to compute the shortest paths from vertex $\boldsymbol{a}$ to all other vertices j in function of the variables edges x^1 *and* x^2 :

Vertex j	$\boldsymbol{a}$	b	x^1_1	x^1_2	x^2_1	x^2_2	p	q
Compute the shortest paths form a to all other vertices j in function of the variables edges x^1 *and* x^2	0	1	2	Min $(8, 2+x^1)$	Min $(10, 5+x^1)$	Min $(17, 11+x^1, 10+x^2, 5+x^1+x^2)$	Min $(10, 4+x^1, 14+x^2, 9+x^1+x^2)$	Min $(18, 12+x^1, 11+x^2, 6+x^1+x^2)$

We call *critical conditions* of a given vertex, the set of length functions associated to the four paths to this given vertex computed by the algorithm. Because the functions of this set are constant, or very simple linear functions, they can be computed and compared very easily. Thus for each target vertex, a set of a maximum of four alternative paths can be stored along with the associated set of critical conditions. As soon as any variable weight changes, the critical conditions of the target vertex just need to be recomputed and compared. Then the new shortest path may be choosen among the alternative paths stored for this vertex. No recomputation of shortest paths is needed, no data beside the current values of the variable arcs need to be exchanged, and all decisions may be taken locally. The complexity of the algorithm itself is $O(n^2)$.

The proposed algorithm can be used to build alternative routing tables in a computer network, or for the routing or re-routing of a truck in a truck delivery problem.

In the future, we intend to work on extending this result to non directed graphs, and to the computation of longest paths for scheduling problem.

Références

1. R. K. Ahuja, T. L. Magnanti, J. B. Orlin, Network Flows: Theory, Algorithms, and Applications, PP.164-165, Prentice Hall 1993.

2. M. Alivand, A.A. Alesheikh, M.R. Malek, "New method for finding optimal path in dynamic networks". World Applied Sci. J.,2008, 3: 25-33, 2008.

3. H.R. Bajgan, R.Z. Farahani. "Using colony system and neighborhood search for dynamic vehicle routing problem". American Journal of Operational Research, vol. 2, no.4, pp. 31-44, 2012.

4. D.J. Bertsimas. Probabilistic combinatorial optimization problems. PhD thesis, Massachusetts, Institute of Technology, 1988.

5. S. Balev, F. Guinand, Y. Pigné, "Maintaining Shortest Paths in Dynamic Graphs". In proceedings of the International Conference on Non-Convex Programming: Local and Global Approaches Theory, Algorithms and Applications (NCP'O7). 2007, December 17-21, Rouen, 2007.

6. Nicolas Boria, Vangelis Th. Paschos, Optimization in dynamic environments, CAHIER DU LAMSADE 314, Université Paris-Dauphine, Novembre 2011.

7. J-Y Colin, M. Nakechbandi, A.S., Ould Cheikh, "Searching Shortest Paths in Weakly Dynamic Graphs", ECCS'12 : European Conference on Complex Systems, sept. 3-7 2012, Brussels 2012.

8. C. Demetrescu and G.F. Italiano, "A new approach to dynamic all pairs shortest paths", J. ACM, 2004, 51(6):968–992, 2004.

9. D. Fulkerson, Expected critical path lengths in PERT networks. Operations Research 10 808 - 817, 1962.

10. P. Jaillet. Probabilistic traveling salesman problems. PhD thesis, Massachusetts Institute of Technology, 1985.

11. H. Mao, "Pathfinding Algorithms for Mutating Graphs", Computer Systems Lab 2007-2008.

12. A. S. Ould Cheikh, J-Y Colin, M. Nakechbandi, "Problème de transport dans un graphe faiblement dynamique", GOL'12 : 1st International IEEE Conference on Logistics Operations Management, 17-19 October, Le Havre, 2012.

13. G. Ramalingam, T. Reps. "On the computational complexity of dynamic graph problems", Theoret. Comput. Sci. 1996, 158 233-277, 1996.